(12) UK Patent Application (19) GB (11) 2 253 079 (13) A

(43) Date of A publication 26.08.1992

(21) Application No 9125108.2

(22) Date of filing 26.11.1991

(30) Priority data
(31) 55391 (32) 19.02.1991 (33) IE

(51) INT CL[5]
G06F 12/14

(52) UK CL (Edition K)
G4A AAP

(56) Documents cited
GB 2237666 A GB 2059652 A GB 1329721 A

(58) Field of search
UK CL (Edition K) G4A AAP
INT CL[5] G06F 12/14

(71) Applicant
Tolsys Limited

(Incorporated in Ireland)

Innovation Centre, Trinity College, College Green, Dublin 2, Ireland

(72) Inventors
Brian Arthur Coghlan
Jeremy Owen Jones

(74) Agent and/or Address for Service
Marks & Clerk
57-60 Lincoln's Inn Fields, London, WC2A 3LS, United Kingdom

(54) Stable memory protection using capability tables

(57) A stable memory circuit includes a capability circuit 10 which has an object capability table 14 and a process capability table 15 each of which are stored within stable memory banks. The object capability table 14 stores a capability entry 16 for each page of the stable memory banks and the process capability table 15 stores a list of the object capabilities for those pages that a process may validly access. Since control and access of the capability tables takes place within the stable memory circuit the host processor is unaware of their values and thus it is impossible to modify any capabilities deliberately or accidently. The capabilities may also be used to provide central update logging and central dependency logging. A cache circuit incorporating a stable memory circuit which serves both the data and tags for the cache is disclosed. Further, stable memory circuits may also include disk and network interfaces and may be arranged in fully or partially instantiated N-dimensional configurations.

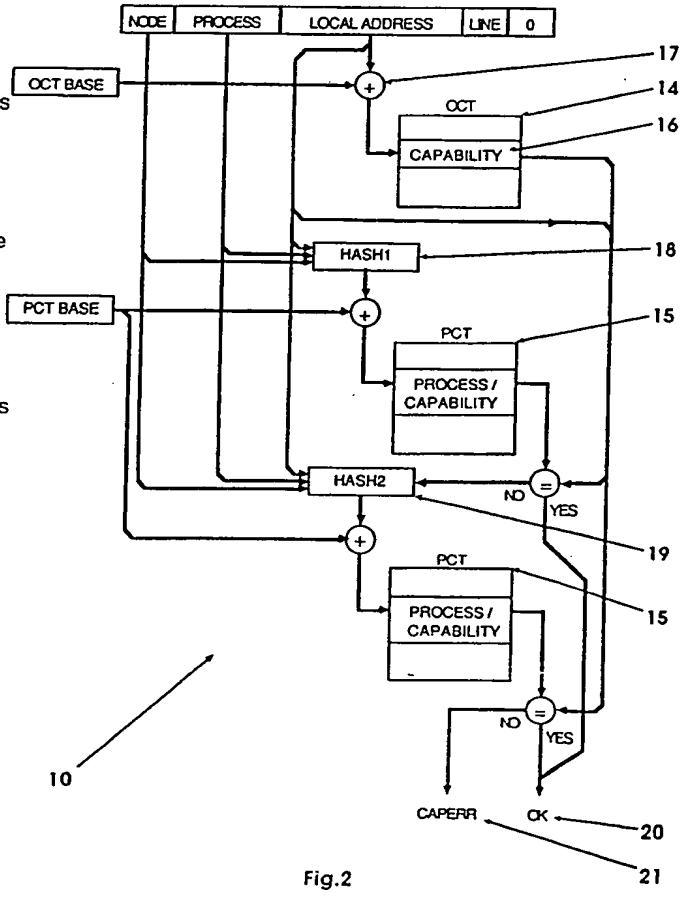

Fig.2

GB 2 253 079 A

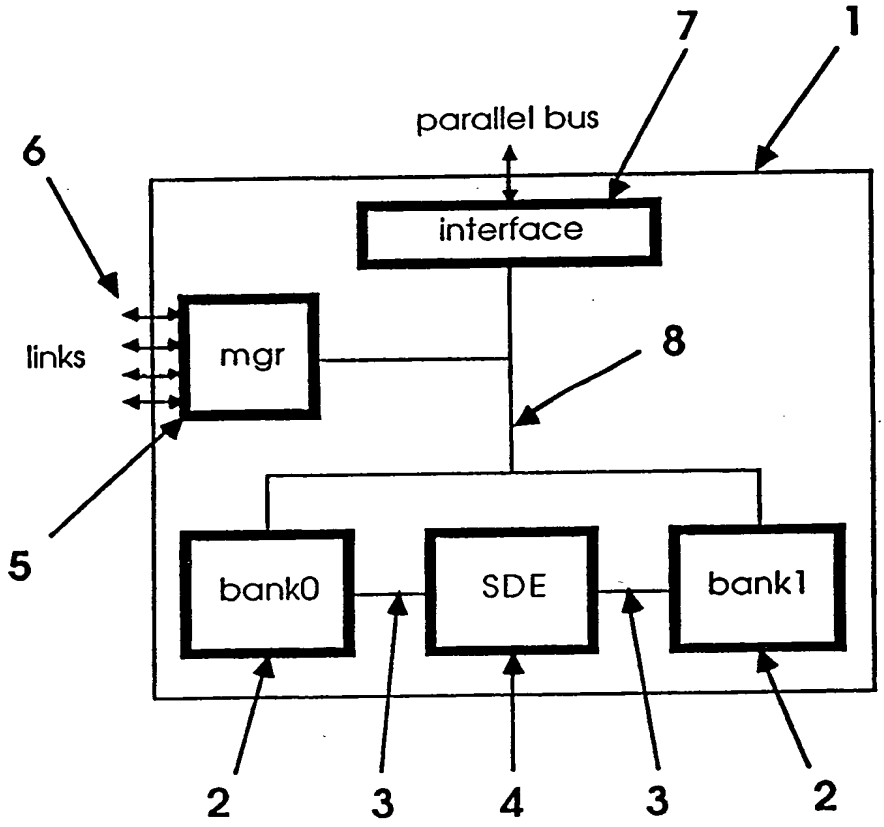

Fig.1

NSDOCID: <GB 2253079A>

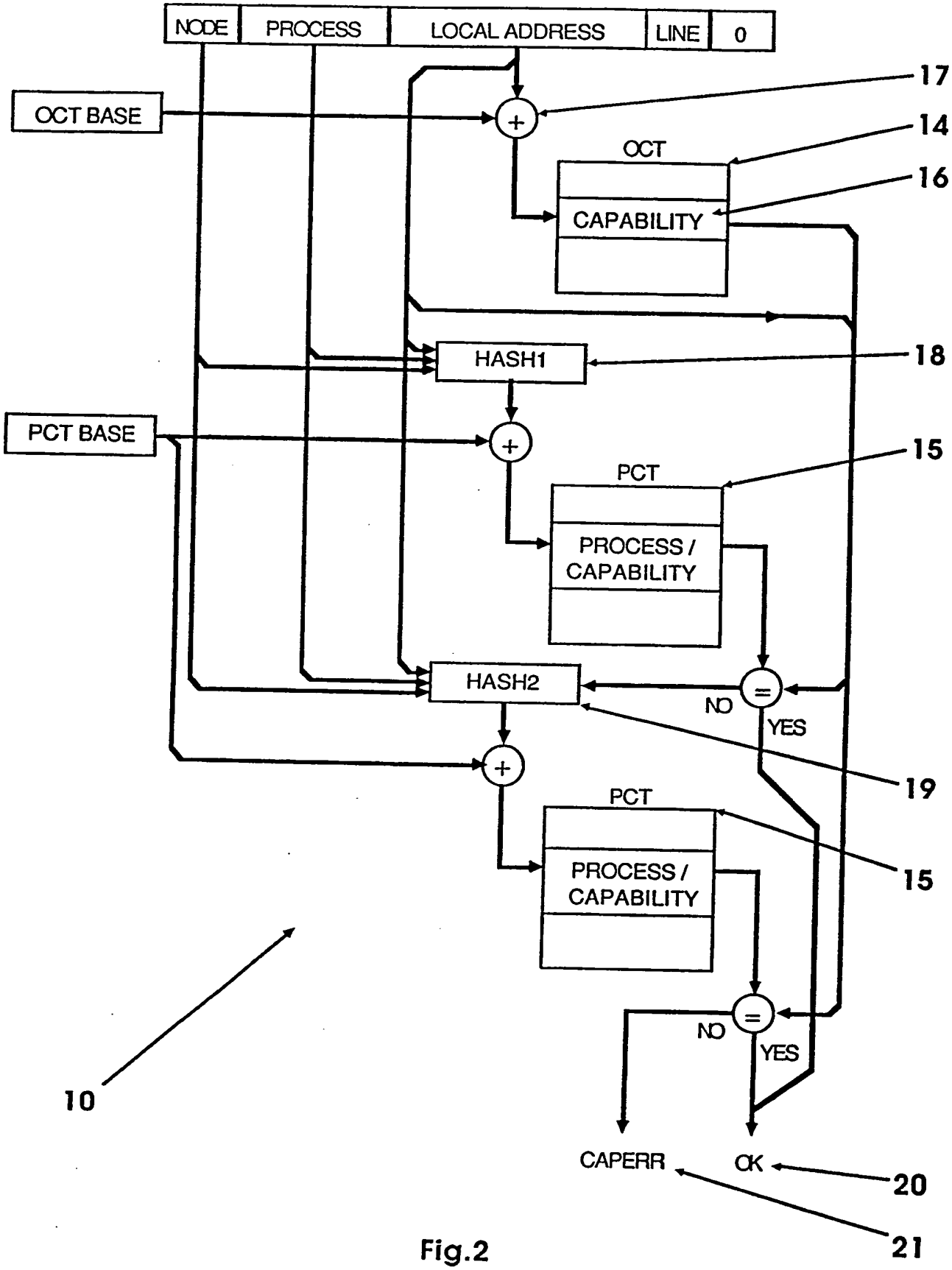

Fig.2

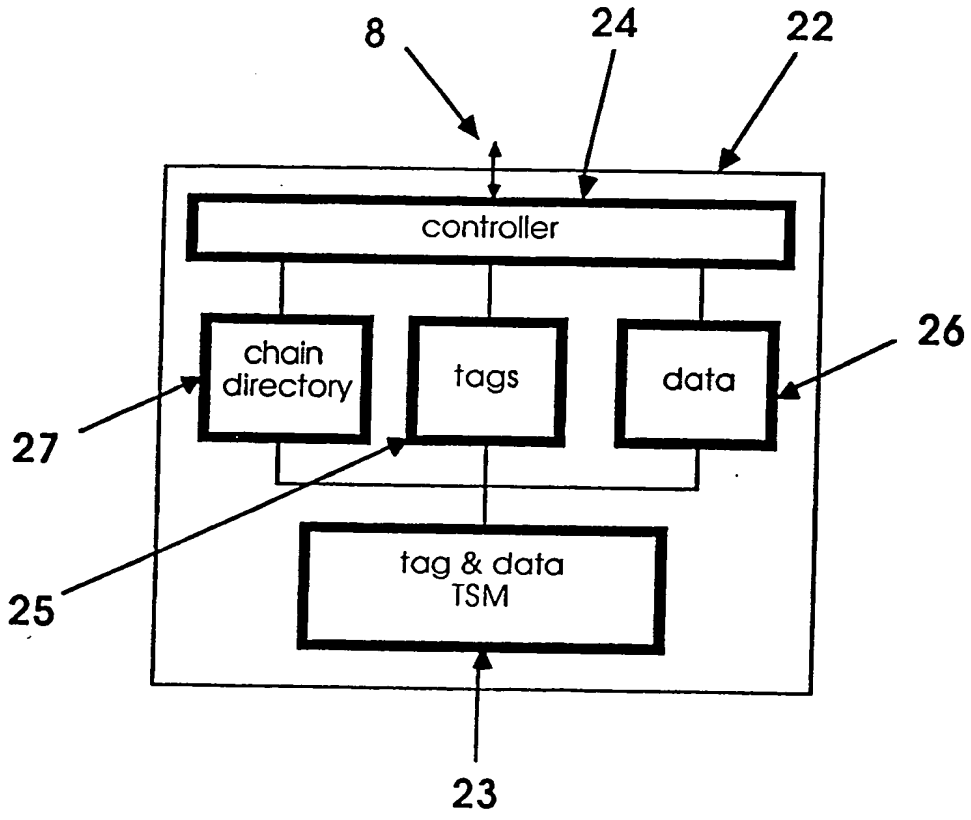

Fig.3(a)

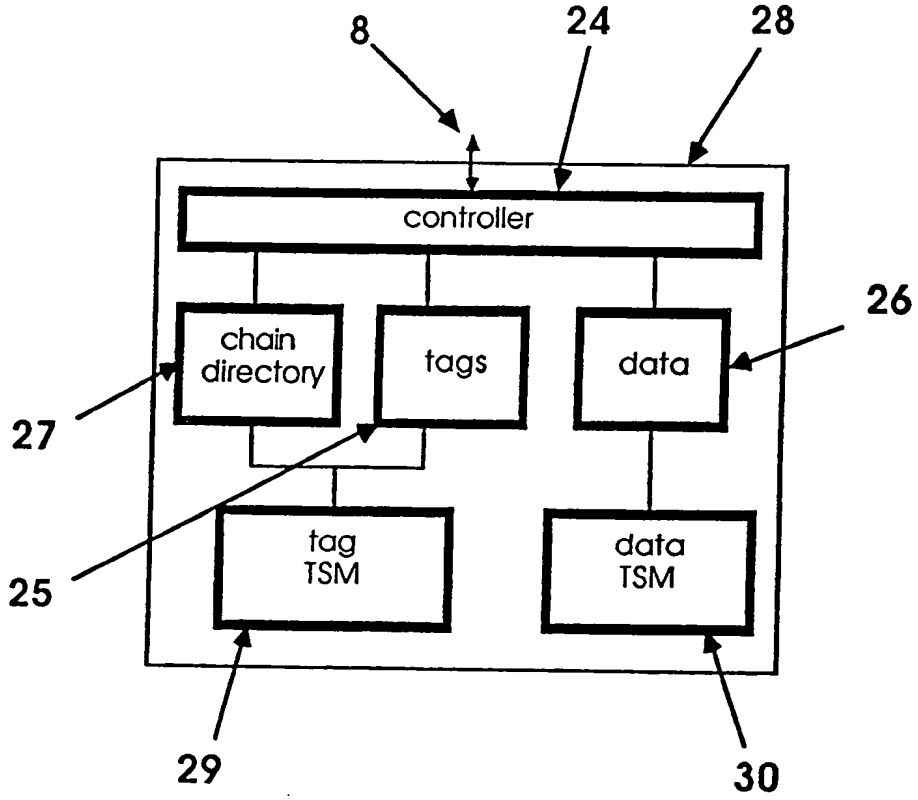

Fig.3(b)

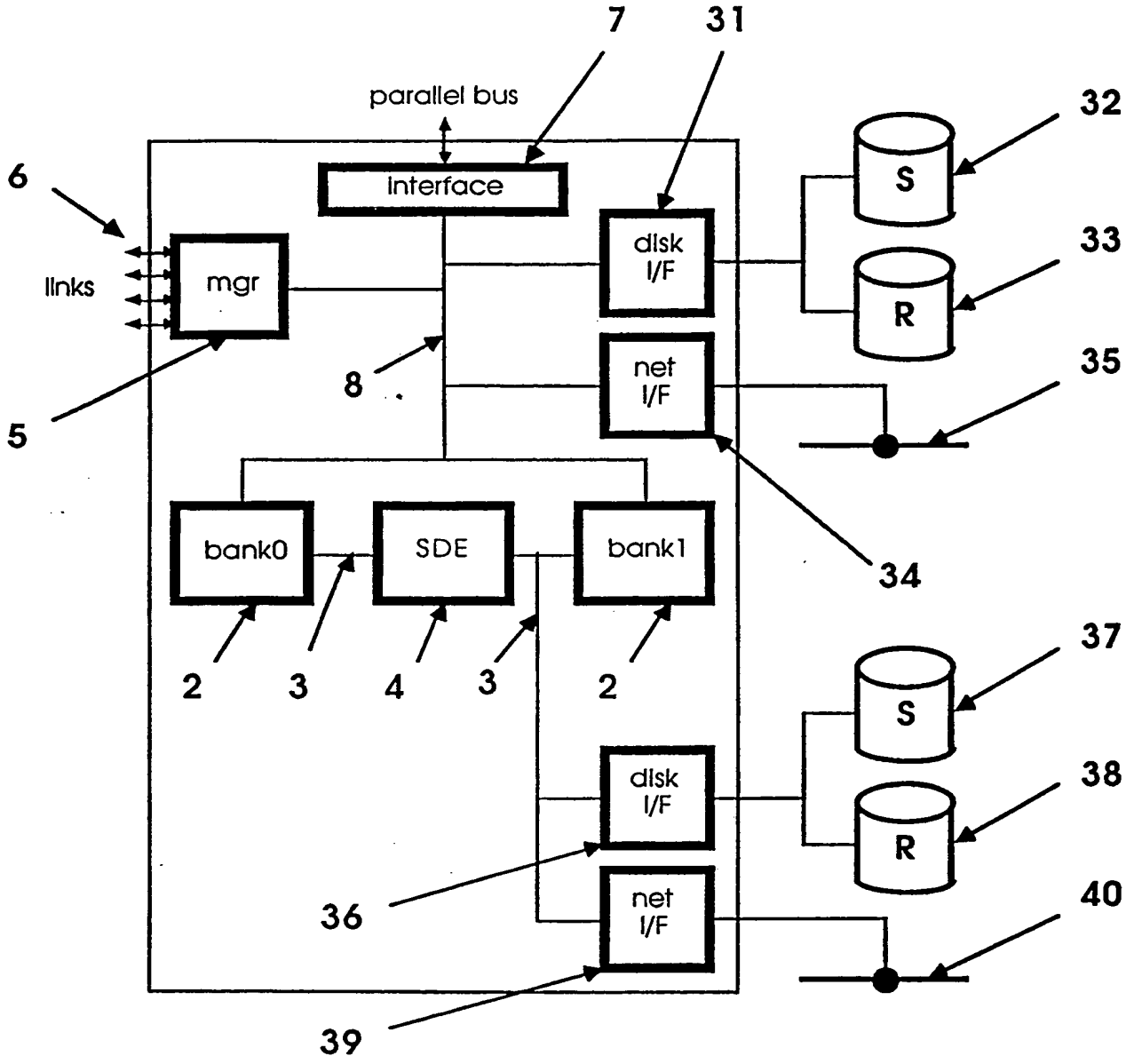

Fig.4

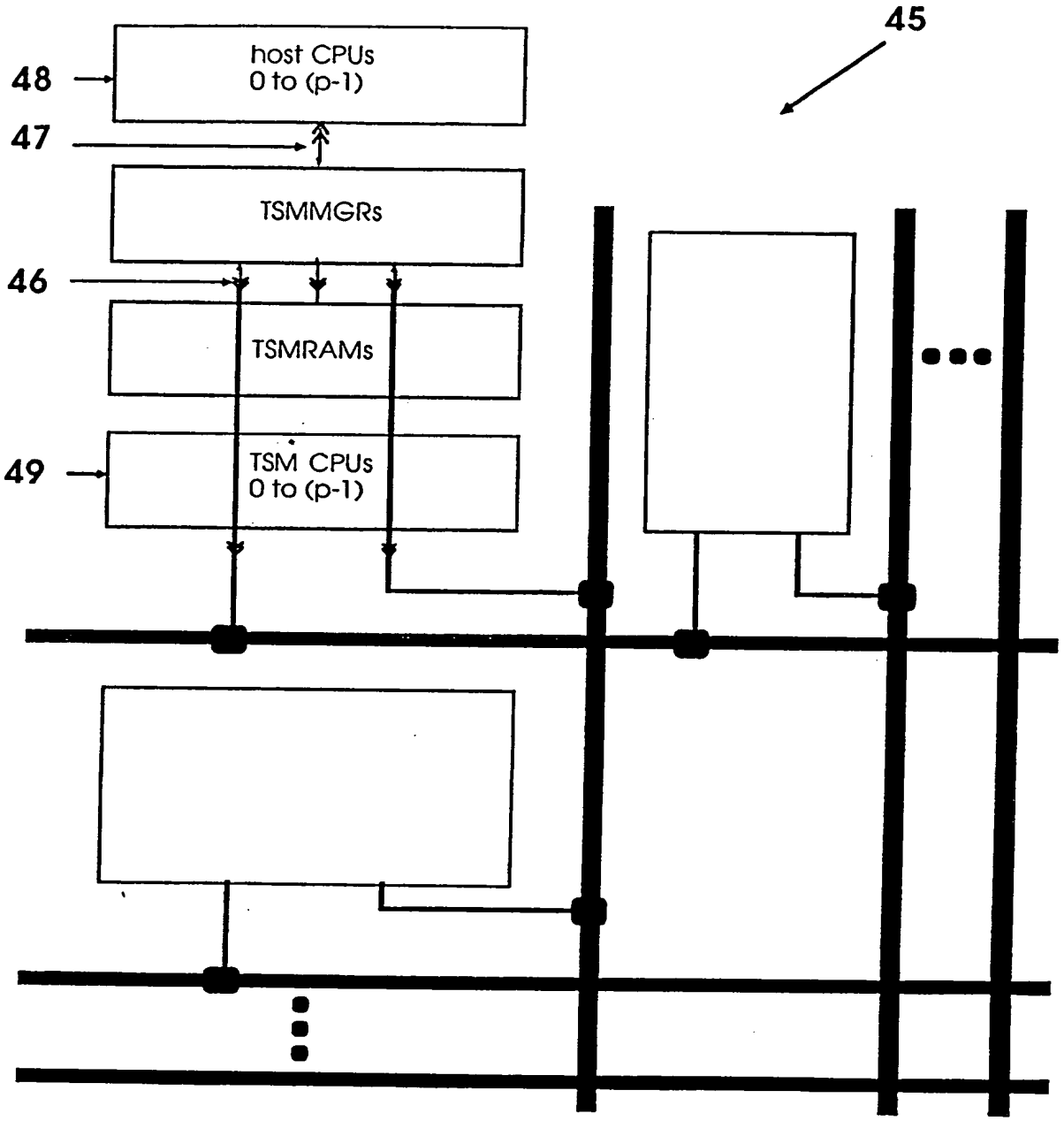

Fig.5

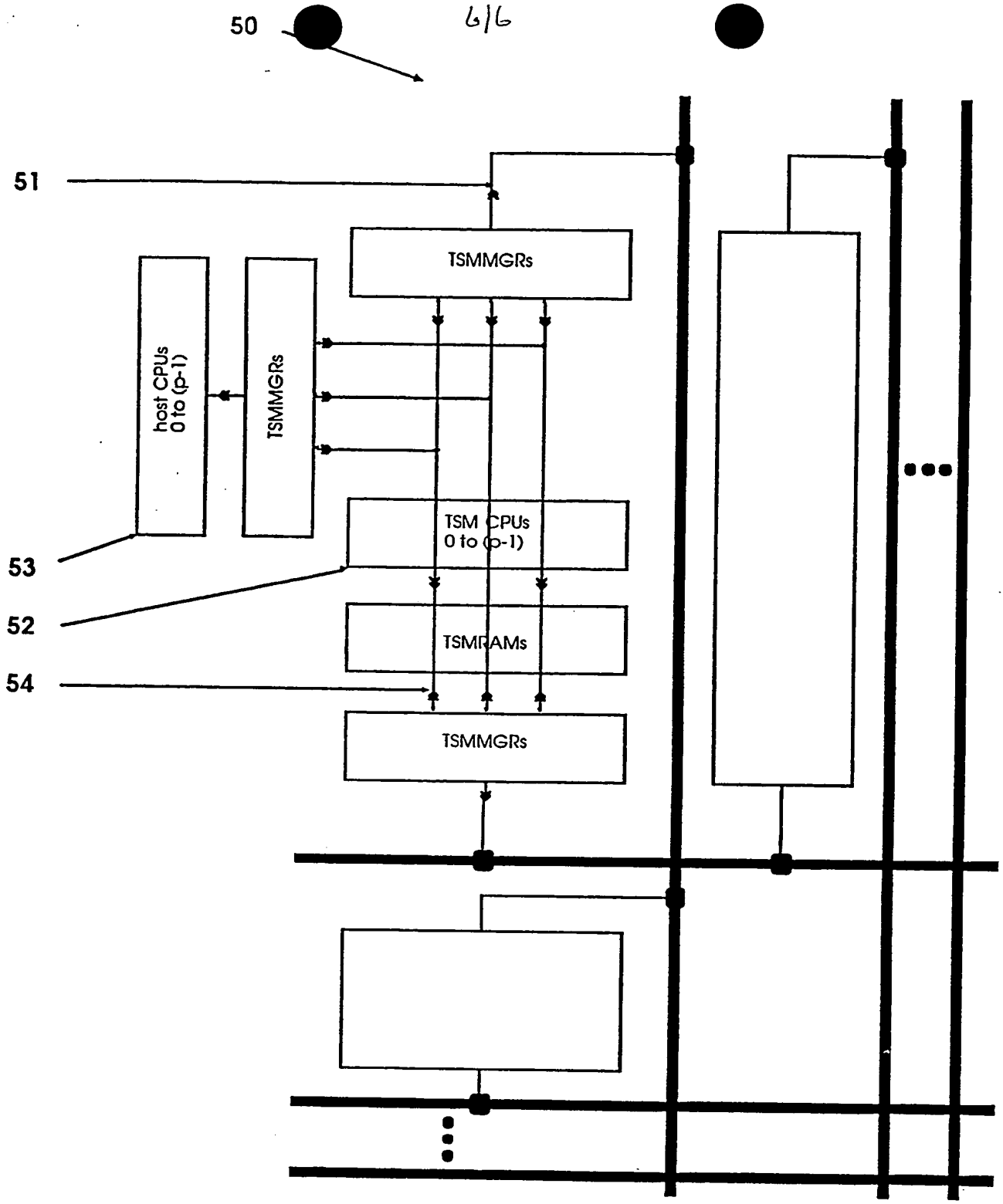

Fig.6

"Improvements in and relating to stable memory circuits"

The present invention relates to stable memory circuits such as that described in British Patent Specification No. 2237666A.

In this specification, the phrase "stable memory circuit" is intended to cover a circuit which includes not only banks of stable memory but also control and manager circuits associated with the stable memory banks.

The present invention is directed towards providing a stable memory circuit with improved protection, caching, disk and network support, and fully or partially instantiated N-dimensional configurations for performing various functions.

According to the invention, there is provided a stable memory circuit comprising means for storing capabilities indicating protection for and function of portions of stable memory banks of the circuit.

Ideally, there is capability for each portion of a memory bank and a means for indicating the capabilities for those portions

that each process or activity associated with the stable memory circuit may access (i.e. which capabilities it "owns").

In one embodiment, there is an object capability table and a process capability table, the object capability table specifying a capability entry for each page of the memory bank, the process capability table storing object capabilities for those pages which may be accessed by each process.

The invention also provides a method of accessing a portion of a memory bank in a stable memory circuit comprising the sub-step of accessing the capability tables to determine whether or not access is allowed to that portion of the memory bank.

The invention also provides methods for using the capabilities to collect lists of those pages modified by each process, and to track and resolve dependencies between external caches.

Detailed Description of the Invention

The invention will be more clearly understood from the following description of some embodiments thereof, given by way of example only with reference to the accompanying drawings in which:-

Fig. 1 is a representation of a stable memory circuit of the prior art;

Fig. 2 is a representation of one embodiment of the capability mechanism of a stable memory circuit of the invention;

Figs. 3(a) and 3(b) are representations of stable cache circuits incorporating a stable memory;

Fig. 4 is a representation of a stable memory circuit including interfaces for peripheral devices;

Figs. 5 and 6 are views showing network arrays incorporating stable memory circuits.

Referring to the drawings, and initially to Fig. 1 there is illustrated a stable memory circuit such as that described in British Patent Specification No. 2237666A. The stable memory circuit 1 comprises two or more stable memory banks 2 made up of dual-ported memory circuits such as VRAMs. The memory banks 2 are interconnected by serial data paths 3 in which is connected a stable data engine 4. The stable memory circuit 1 also includes a manager circuit 5 having serial links 6, and a parallel bus interface 7 connected to a parallel bus 8.

Whilst the protection of stable memory can be supported by simple access flags in the address path, here we describe a more elaborate mechanism, where the protection is supported by "capabilities".

Referring now to Fig. 2, one embodiment of the capability mechanism 10 forming part of a stable memory circuit of the invention is illustrated. This capability mechanism 10 uses two tables, namely, an object capability table 14 and a process capability table 15. The object capability table 14 stores a capability entry 16 for each page of the stable memory bank and the process capability table 15 stores a list of the object capabilities for those pages that a process may validly access. As shown, the object capability table is addressed using the memory address. A hash function 18 is applied to the memory address, process identifier, and node identifier in the addressing of the process capability table 15, for which re-hashing 19 may be required. An output line 20 is used for signalling that access may validly proceed and an output line 21 is used if an access may not validly proceed, in which case a "Capability Microtrap" may be generated. In this embodiment, the capability tables 14 and 15 are held within the stable memory banks themselves. Each memory board may contain entries relevant to its own memory, thereby allowing for automatic expansion of the capabilities as the capacity of each board is increased or as the number of memory boards is increased. The definition and handling of these tables should be kept entirely distinct from those for virtual memory management schemes.

The capability tables may be cached at two levels. Firstly, since the tables may be extensive, the entries in memory at

any particular time may only be a cached subset of larger tables that may exist on disk. Secondly, to improve performance the parallel bus interface may cache a subset of these memory-resident entries. Parallel bus interface capability cache misses can be automatically resolved by fetching the relevant entry from memory. If no memory-resident entry exists, a Capability Microtrap can be used to interrupt the manager circuit so that it can respond. If the miss indicates a capability entry that is still on disk it may cause the entry to be fetched from disk, otherwise it may handle this as a fault. Faulting on write may be deliberately enforced after every checkpoint so that a list of those pages modified since the last checkpoint may be collected for each process. This is a very important feature, which we call "central update logging", and is an essential component of the "central dependency logging" described later.

The most important feature is that the control and accessing of the capability tables takes place within the stable memory circuit and accordingly the host processor is not aware of their values. In this way, it is impossible to modify any capabilities either deliberately or by accident.

Capability tables define the protection for memory, and imply very little about the stability state of the memory, which is properly maintained within 'stability data structures' for allocated stable memory (stability state is only appropriate

to stable memory). However, capabilities should distinguish between stable and non-stable memory, simply to protect the stable memory areas, and the stability data structures can be merged into the capability tables.

"Stable pages" may be allocated either individually or in clusters. Since a stable page is composed from two or more normal pages, each with its own capability entry, the capability tables for stable pages may be replicated (as stable pages), thereby enhancing fault tolerance. The replication also allows a memory read access of the capability entry to be returned from an alternate bank to that for which the access is intended, although this requires that the address for the alternate bank by derived by such logic that it points to the correct capability entry. These entries may be cached by capability caches and checked by appropriate capability logic at the parallel bus interface, and subjected to error detection and correction in the same way as for data.

The stability data structures and capability entries may initially be set to non-stable by the manager circuit, allowing host processors to consider the stable memory circuit as a normal bank of memory of twice or more the capacity of each memory bank. Stable memory can be allocated on a per-page basis, each stable page occupying the space normally occupied by two or more (not necessarily contiguous) memory pages, thus reducing the total capacity incrementally. When

no more free pages exist, the host processor must surrender the two or more normal pages for every stable page it asks the manager circuit to allocate, and remap the new stable page, implying that only host processors supporting pages virtual memory schemes may efficiently utilise this stable memory.

It will be appreciated that the use of the capabilities of the invention in this way protects the memory banks from failed host processors or software so that existing non-fail-stop processors may be used for low-cost, high-availability applications.

When only a single parallel bus interface is used, additional external write-through or blocking caches will not affect cache coherency but may affect speed, whilst external write-back caches may affect both coherency and speed. Where multiple parallel bus interfaces are used, centralised "chain directories" (for the external caches) for each parallel bus interface can indicate which cache lines are held externally, and hence which accesses to the stable memory circuit must be propagated forward to the relevant external caches attached to that parallel bus interface. Performance may be greatly improved by maintaining these in a chain directory cache at the parallel bus interface.

Checkpointing is more difficult if caches (particularly write-back caches) are used externally to the stable memory circuit, since dependencies between processes will need to be tracked,

and there may be dependencies between data in the various caches that will need to be flushed together. Capabilities can be used to resolve dependencies for external caches by a technique we call "central dependency logging". The cache circuit can 'call-in' the outstanding dirty data from write-back external caches for a checkpoint, either by requesting a flush of all dirty data from the external caches that hold related data (related by dependencies), or by selectively flushing by issuing parallel bus reads for this data (the external caches will then intervene on behalf of memory). The capability mechanism can assist in either of these procedures, since the Capability Microtrap service routines may gather a list of all the modified pages of a process since the last checkpoint (using central update logging), the Process Capability Tables may indicate all those external caches and processes that have shared access to those pages, and a centralised chain directory (for the external caches) within the stable memory circuit may indicated which external cache lines in a page may be dirty.

Configurations that benefit from the Invention

The above arrangements (based on capabilities) provide strong support for caching in a system that uses stable memory. They may, however, also be configured for a cache, yielding a "transport stable cache".

Referring now to Figs 3(a) and 3(b), cache circuits are illustrated incorporating stable memory circuits of the invention. In Fig. 3(a), there is illustrated a cache circuit 22 incorporating a stable memory circuit 23 which stores both the data and tags for the cache. The cache circuit 22 includes a cache controller 24 connected to a tag cache 25, a data cache 26 and an chain directory cache 27.

In Fig. 3(b), there is illustrated a cache circuit 28 having a stable memory circuit 29 which stores cache tags and a separate stable memory circuit 30 which stores cache data. Each of the cache circuits 22 and 28 could be regarded as a "transparent stable cache" as the stable memory operations would not be monitored or visible to a host processor. Because tags are stored in the tag cache 25 for both cache circuits, a high probability exists that the tags may be found in the tag cache without having to recall them from the associated stable memory circuit. If the tag cache is a write-through cache, then the cache controller 24 may update the tags at the tag cache 25 and leave the tag cache 25 to update the stable memory circuit 23, or 29, as the case may be. Similarly, the stable memory circuit 23 or 30, is supplemented by the write-through or blocking data cache 26, which will not affect cache coherency but may affect speed.

The cache circuits may be used as the only memory in the computing system which allows all system memory, including

caches, to be stable. This simplifies the allocation and management of stable memory, as well as simplifying recovery after a fault.

With it's strong support for caching, a stable memory circuit using the capabilities of the invention may form the core of the nodes of a fully or partially instantiated N-dimensional array network computing system, mapped into a global address space and supporting a system caching mechanism that both maintains global cache coherency and minimises the traffic on the system interfaces. An internal write-through or blocking cache, or additional external write-through caches, will not affect cache coherency, but may affect speed. The capabilities may be used to allow external write-back caching.

N-dimensional array networks may be constructed in two ways, both of which are most easily explained for the fully instantiated two-dimensional case where a two-dimensional array of busses will arise. Firstly, referring to Fig. 5 which illustrates a stable memory circuit bus network 45, the busses include internal stable memory circuit parallel busses 46 (also 8), with a parallel bus 47 bridging the array bus intersections. Host processors 48 are attached to the intersection busses or stable memory circuit processors 49 are attached to the array busses. Although stable memory is

attached to the array busses, it is associated with the node at the intersection of these array busses.

Referring now to Fig. 6, a parallel bus network 50 is illustrated in which the roles of the stable memory circuit and parallel buses are swapped so that the array buses are the parallel busses 51 with each stable memory circuit bridging the array bus intersections. Either stable memory circuit processors 52 are attached to the intersection busses, or host processors 53 are attached to the intersection busses via additional manager circuits, or host processors attached to the array busses. Since stable memory is attached to the intersection busses, it is naturally associated with the node at the intersection of the array busses.

This is a more expensive arrangement than that shown in Fig. 5 as every node requires at least two manager circuits instead of one. However, tolerance to faults increases, since each node has duplicated intersection busses, and so can survive a failure of any component relating to one of those busses. Moreover, the capability mechanism at the parallel bus interface allows the stable memory to be better isolated against array bus failures, particularly those that might propagate random behaviour. For these reasons the parallel bus network might be better suited for highly reliable systems. Either form will give the same functionality. Because the stable memory attaches to the array busses, the

stable memory circuit bus network will result in much greater array bus traffic. This need not be critical, since all memory does not need to be stable, and non-stable memory may be attached to the intersection bus to reduce the usage of array bus bandwidth. To extend the array from two to N dimensions, more manager circuits must be added, thereby increasing the number of array busses. Anyone of ordinary skill in the art will recognise that most other network topologies such as hierarchical networks, can be formed from a partially instantiated N-dimensional array network.

In addition to fully and partially instantiated N-dimensional array networks constructed in the above fashion, the stable data paths may be connected in a fully or partially instantiated N-dimensional array network, with a plurality of stable data engines bridging a plurality of stable data paths, each stable data engine connecting to a plurality of memory banks.

In one embodiment that has been constructed, one dimension of a two-dimensional array of stable data paths interconnects stable data engines that connect to two memory banks in the other dimension. Again, anyone of ordinary skill in the art will recognise that most other network topologies can be formed from a partially instantiated N-dimensional array network.

Since transfer of information with a stable memory circuit occurs via the stable data paths, while normal random accesses to the memories can continue to occur concurrently (ie, transparently), the stable data paths can be intercepted or observed by a stable data engine configured in specific ways, for example, configured as a 'vector engine' to allow trivial and non-trivial vector-related functions to be applied to the stable memory data. Trivial functions can include summing or scaling a vector in one bank, and non-trivial functions can include FIR and IIR filtering functions, or for example, the computation for the response of a neuron (as a vector multiply-accumulate). Various other configurations are also described below.

A more important use of the stable data paths relates to disk and network interfaces. Referring now to Fig. 4, there is illustrated a stable memory circuit, in which parts similar to those described with reference to Fig. 1, are identified by the same reference numerals. The stable memory circuit in Fig. 4 includes a disk interface 31 connected to silicon disks 32 and to rotating disks 33. The parallel bus 8 is also connected to a network interface 34 for a network 35. The stable data path 3 is connected to a disk interface 36 for silicon disks 37 and rotating disks 38. The stable data path 3 is also connected to a network interface 39 for a network 40. Note also that the interfaces 32/33 and 37/38 may be combined such that the one disk and network interface is

connected to both the parallel bus 8 and the stable data path 3. The stable memory circuit described in Fig. 4 may be referred to as "transparent stable disk".

It will be appreciated that the connection of the stable data paths to disks and to a network yield very high bandwidth transparent disk and network interfaces. Accordingly, operations such as network communications may be performed by the memory itself and may be exploited in any of the ways that a stable memory circuit may be. Multiple disk and network interfaces can provide redundancy for fault tolerance.

This allows disks to be invisible to a host processor, resulting in lower processor overhead and more efficient utilisation of disks, especially rotating disks. Regarding the rotating disks, the disk drive head will generally be operated in a more efficient manner when controlled via the stable memory circuit as sectors of the disk can be written in sequence. This can be done by aggregating disk writes in a buffer in stable memory, and writing these to disk only when absolutely necessary, and even then writing in large blocks at the end of a log - the disk is written progressively from the first track to the last. This increases system performance. In previous memory-based logging file systems the memory was not fault-tolerant, but in this case, since the log is supported by the stable memory circuit, it is resilient to failures, and hence it is safe to keep some committed data

only in the log. Again it must be appreciated that the use of the capabilities of the invention protects the memory banks and disk and network interfaces from failed host processors or software so that existing non-fail-stop processors may be used for low-cost, high-availability solutions.

The stable memory circuit may also be configured to support both generation-based and copying 'transparent garbage collectors', and indeed for an asynchronous garbage collection of directed graphs of heap allocated storage for functional or object based language environments. In cases where the locations of references to allocated memory objects are discernable to the stable memory circuit, it can then trace these asynchronously, identifying and collecting garbage, and optionally compacting the heap region as a result. The capabilities may be used to assist in this, particularly where each referenced object has a different capability.

For stable memory circuit systems with only two banks of memory any such garbage collection is only really efficient for large objects, which can be collected transparently using the stable memory circuit block copying mechanisms. If, however, more than two banks of memory are available, and if any one of these can be exclusively allocated to the Garbage Collector (either the manager or some specialised devise) at any time, then the heap may be (transparently) block copied to this bank, where the Garbage Collector can (transparently)

identify and collect garbage, and optionally compact the heap. When finished, either the bank can simply be reallocated for access by the host processors, or the heap can be (transparently) block copied back to whence it came; either way the result is a new heap. A post process will then need to gather any intervening heap changes and update the new heap accordingly. Note that if concurrent usage of the 'old' heap is not required then the bank holding the heap can simply be allocated to the Garbage Collector, and no post processing will be required. The capabilities may then be adjusted to deny unwanted accesses while the garbage collection is in progress.

In the same vein, consistency (within a database) is often maintained by atomic commit operations using Stable Storage, where performance is limited by the disk characteristics. Ultimately this limits the number of Transactions-Per-Second (TPS), generally considered a vital figure of merit for database systems. A stable memory circuit can be configured to perform fast and fault tolerant transparent atomic commits by buffering updates in stable memory, giving dramatically increased TPS.

The stable memory circuit can be configured as a buffer pool for database segments and persistent memory. Such segments can be mapped on demand by the stable memory circuit if it intercepts accesses by segments by translating a segment name

to an appropriate position in the buffer pool. Active segments may be pinned while in use, and unpinned segments may be flushed. The capabilities can be used to provide efficient support for this mechanism by converting segment names to buffer pool addresses and managing the buffer pool. Since the buffer is supported by the stable memory circuit, it is resilient to failures, and hence it is safe to keep some persistent data only in the buffer.

In the same way, the stable memory circuit can be configured to provide efficient support for log based recovery for recoverable memory by reducing disk writes for frequently committed data. The stable memory circuit can be organised as a stable cyclic buffer log, in which multiple versions of the same data may be found, the most recent of which is near the front of the log. Committed data items may be flushed from the log asynchronously, omitting disk writes for data for which there is a later committed version in the log, thus saving disk writes.

All these configurations are materially assisted by the capabilities of the invention, which also protects against failed host processors or software so that existing non-fail-stop processors may be used for low-cost, high-availability solutions.

The invention is not limited to the embodiments hereinbefore described but may be varied in construction and detail.

CLAIMS

1. A stable memory circuit comprising means for storing capabilities indicating protection for and function of portions of stable memory banks of the circuit.

2. A stable memory circuit as claimed in claim 1, wherein there is a capability for each portion of a memory bank and a means for indicating the capabilities for those portions that each process or activity associated with the stable memory circuit may access.

3. A stable memory circuit as claimed in claims 1 or 2, comprising an object capability table and a process capability table, the object capability table specifying a capability entry for each page of the memory bank, the process capability table storing object capabilities for those pages which may be accessed by each process.

4. A method of accessing a portion of a memory bank in a stable memory circuit comprising the step of accessing a capability table to determine whether or not access is allowed to that portion of the memory bank.

5. A method called central update logging, for using the capabilities to collect a list, for each process, of those pages modified by that process.

6. A method, called central dependency logging, for using the capabilities to track and resolve dependencies between caches that are external to the stable memory circuit.

-21-

Patents Act 1977
Examiner's report to the Comptroller under Section 17 (The Search Report)

Application number

9125108.2

Relevant Technical fields

(i) UK Cl (Edition K) G4A (AAP)

(ii) Int Cl (Edition 5) G06F 12/14

Search Examiner

MISS A C CLARKE

Databases (see over)

(i) UK Patent Office

(ii)

Date of Search

25 FEBRUARY 1992

Documents considered relevant following a search in respect of claims

1 TO 4

Category (see over)	Identity of document and relevant passages	Relevant to claim(s)
Y,P	GB 2237666 A (PROVOST) Whole document	1,4
Y	GB 2059652 A (PLESSEY) Page 1 lines 70-75	1,2,4
Y	GB 1329721 A (PLESSEY) Page 3 lines 66-78	1,2,4

SF2(p)

bbb - c:\wp51\doc99\fil000040

Category	Identity of document and relevant passages	Relevant to claim(s)

Categories of documents

X: Document indicating lack of novelty or of inventive step.

Y: Document indicating lack of inventive step if combined with one or more other documents of the same category.

A: Document indicating technological background and/or state of the art.

P: Document published on or after the declared priority date but before the filing date of the present application.

E: Patent document published on or after, but with priority date earlier than, the filing date of the present application.

&: Member of the same patent family, corresponding document.

Databases: The UK Patent Office database comprises classified collections of GB, EP, WO and US patent specifications as outlined periodically in the Official Journal (Patents). The on-line databases considered for search are also listed periodically in the Official Journal (Patents).

Published 1992 at The Patent Office. Concept House. Cardiff Road. Newport. Gwent NP9 1RH. Further copies may be obtained from Sales Branch. Unit 6. Nine Mile Point. Cwmfelinfach. Cross Keys. Newport. NP1 7HZ. Printed by Multiplex techniques ltd. St Mary Cray. Kent.

<SDOCID: <GB 2253079A >

THIS PAGE LEFT BLANK